Structured Feedback for Preference Elicitation in Complex Domains

Stefano Teso
University of Trento
Trento, Italy
teso@disi.unitn.it

Paolo Dragone
University of Trento
Trento, Italy
paolo.dragone@unitn.it

Andrea Passerini
University of Trento
Trento, Italy
passerini@disi.unitn.it

Abstract

Preference elicitation is concerned with inferring a latent utility function from user feedback. Typical elicitation approaches iteratively query the user about (supposedly informative) pairs of candidate configurations; the collected pairwise preference constraints are used to improve the utility estimate. The ongoing transition from flat to structured configuration spaces is opening promising opportunities for the design of more general and effective feedback mechanisms. In this position paper, we outline a framework, based on a general notion of *structured constraint*, encompassing existing and novel formulations.

1 Introduction

Preference elicitation [Goldsmith and Junker, 2009] is a key component of personalized recommendation and decision support systems. Queries are presented to the user according to a certain query selection strategy [Chajewska *et al.*, 2000], and the feedback received is used to refine the current approximation of the user utility. The process is repeated until the user is satisfied with the recommendation. Classic preference elicitation methods have been concerned with pairwise preference queries: the learner proposes two candidate configurations to the user, who replies by marking one configuration as more desirable than the other. However, when configurations are *structures*, i.e. configurable entities made of components and relationships between them, more complex feedback mechanisms become viable.

In this position paper we untangle different kinds of feedback strategies for structured configurations and position them within a systematic elicitation framework, encompassing preference elicitation from pairwise constraints [Chajewska *et al.*, 2000], coactive learning [Shivaswamy and Joachims, 2015], conversational (or example-critique) recommender systems [Chen and Pu, 2012], as well as constructive methods [Teso *et al.*, 2016]. In all of these, user feedback can be cast in terms of constraints that affect different aspects of the learning process. For instance, pairwise preferences constrain (or bias) the space of parameters. Other kinds of user-provided constraints can be devised, affecting instead the space of configurations (either at the attribute or feature level) and the space of utility functions. We present different classes of constraints, which provide complementary information to the learner, and (crucially) can not always be converted into one another.

In order to ground the discussion, we next present three prototypical *constructive* scenarios, where the goal is to construct potentially novel items, and the set of candidates is defined by constraints rather than enumerated explicitly. In these examples, user feedback does not fall in the pairwise preference schema, but can be naturally interpreted as (combinations of) diverse constraints.

Example 1: commenting recipes. A robotic bartender is tasked with generating cocktail and food recipes tailored toward the tastes of a specific user. Configurations amount to proportions of specific ingredients. The interaction model involves the robot proposing a candidate food/drink to the user, who comments about aspects that she does not like: one or more ingredients may be harmful (because of food allergies or intolerances), the dish may be lacking in some respect (not sweet enough or too spicy) or may possess clashing aspects (simultaneously sweet and salty). In this setting, the feedback provides constraints about individual attributes and compound features alike.

Example 2: annotating sketches. An automated design support system constructs optimized furniture layouts for the living place of a user. A layout can be viewed as a hybrid structure, including categorical and numerical attributes and relations between them. Upon seeing a sketch (or rendering) of a proposal, the user replies by annotating portions of the sketch with critiques such as: "the TV set is too close to the kitchen door", or "the bathroom is too far away from the bedrooms". Here the feedback is not about the whole configuration, but rather about sub-structures and their relations.

Example 3: understanding ergonomics. An on-line service sells personalized armchairs manufactured with 3D printing technology, and leverages a recommender system for probing the client's preferences. Upon viewing a newly constructed item, the user submits a full-fledged review to the website. The review includes comments on very definite aspects (colors and materials) to more volatile ones (aes-

thetics and ergonomics). Crucially, the relation between attributes/features and the latter is difficult to define even to domain experts, let alone encoded into a mathematical model.

2 Learning with Structured Feedback

Following common practices, we distinguish between three different spaces. A structured configuration x is composed of categorical and numerical attributes; the space of feasible configurations $\mathcal{X}$ is implicitly or explicitly defined by constraints. The learner has access to some feature representation $\boldsymbol{\phi}(x) \in \Phi$; individual features may be Booleans, integers or reals. Utility functions, taken from a set $\mathcal{U}$, establish a total or partial order over the feature space. Given an (unobserved) utility function $u^* \in \mathcal{U}$ encoding the true user preferences, the goal of the learner is to find a function $u \in \mathcal{U}$ incurring small loss.

We assume the learning procedure to be laid out as in Algorithm 1; many existing methods can be readily related to this template. At each iteration, a high utility configuration x is chosen in step 4 and presented to the user. This step is natively constructive: when $\mathcal{X}$ is not ground, the maximization is free to discover entirely novel configurations. Feedback collection takes place in step 5, where the user reply is converted to a set of *constraints* which are added to the current set $\mathcal{C} := \mathcal{C}_u \cup \mathcal{C}_x \cup \mathcal{C}_{\boldsymbol{\phi}}$. In step 6, the utility estimate u is updated accordingly. Optionally, the feature function $\boldsymbol{\phi}$ is modified in step 7 by augmenting or pruning the feature set.

Utility constraints $\mathcal{C}_u$ serve to favor certain utility functions, for instance the ones consistent with the collected pairwise rankings, in step 6. Our formulation additionally allows $\mathcal{C}$ to also have significantly diverse effects. *Attribute constraints* (as in Example 1) and *sub-structure constraints* (Examples 1 and 2), collectively referred to as $\mathcal{C}_x$, both affect the feasible configuration space, for instance by tying the value of sets of attributes. Finally, *feature transformations* $\mathcal{C}_{\boldsymbol{\phi}}$ (Example 3) guide the update of the feature representation in step 7.

The complexity of the algorithm depends on the choices made on the features $\boldsymbol{\phi}(x)$, utility u, loss function and the constraints $\mathcal{C}$. If all these components depend linearly on x, steps 4 and 6 can be tackled in terms of either Mixed Integer Linear Programming (MILP) [Wolsey and Nemhauser, 2014] or Optimization Modulo Theories (OMT) [Sebastiani and Tomasi, 2015].

Clearly, the choice of constraint types is domain-dependent, and deeply influences the complexity of the two optimization problems. As an example, linear constraints over non-convex features amount to non-convex optimization over the attributes in step 4, rendering exact optimization problematic. The effect is particularly critical for feature transformations (step 7), which imply a much larger search space, and should be used sparingly. We observe however that offline MILP solvers are often fast enough for preference elicitation tasks [Teso *et al.*, 2016], and that incremental optimization methods (e.g. incremental OMT [Sebastiani and Tomasi, 2015]) are perfectly suited for leveraging the iterative nature of the elicitation process.

Algorithm 1 Template algorithm for preference elicitation: $\mathcal{U}$ (resp. $\mathcal{X}$) is the feasible space of utility functions (resp. configurations)

1: **procedure** ELICIT($\mathcal{U}, \mathcal{X}, \boldsymbol{\phi}$)
2: Initialize $u \in \mathcal{U}, \mathcal{C} \leftarrow \emptyset$
3: **for** $t = 1, \ldots, T$ **do**
4: $x \leftarrow \operatorname{argmax}_{x \in \mathcal{X}} u(\boldsymbol{\phi}(x))$ s.t. $\mathcal{C}_x$
5: $\mathcal{C} \leftarrow \mathcal{C} \cup \text{QUERY}(x)$
6: $u \leftarrow \operatorname{argmin}_{u \in \mathcal{U}} \text{loss}(u(\boldsymbol{\phi}(\cdot)), \mathcal{C}_u)$
7: $\boldsymbol{\phi} \leftarrow \text{TRANSFORM}(\boldsymbol{\phi}, \mathcal{C}_{\boldsymbol{\phi}})$
8: **end for**
9: **end procedure**

3 Related Work

Coactive learning (CL) [Shivaswamy and Joachims, 2015; Raman *et al.*, 2013] has been proposed for preference elicitation in structured domains. In CL candidate structures provided by a solver are interactively improved by a domain expert; user modifications are interpreted as ranking constraints. Notably, CL has been extended with limited support for modelling the cognitive cost of queries [Goetschalckx *et al.*, 2015]. Given its reliance on ranking constraints, coactive learning naturally falls within our structured constraint framework.

Another group of related approaches is critiquing-based (or conversational) recommender systems [Viappiani *et al.*, 2006; Chen and Pu, 2012]. In this setting the feedback takes the form of critiques about specific attributes of the proposed configurations. Collected feedback is used to restrict the set of candidate items, in a similar manner to constructive preference learning; current approaches however expect the item pool to be defined extensively.

Conditional preference networks [Boutilier *et al.*, 2004] are a related combinatorial model for expressing preference relations. Most works in this area however do not consider the problem of online CP-network learning. The few that do (e.g. [Koriche and Zanuttini, 2010; Guerin *et al.*, 2013]), avoid the issue of query selection methods, and do not aim at minimizing the number of queries posed to the user. As such, CP-networks are not suitable for interactive preference elicitation.

From an interactive optimization viewpoint, our task is related to the problem of constraint acquisition in constraint solving. In *passive* acquisition [Bessiere *et al.*, 2005; Lallouet *et al.*, 2010; Beldiceanu and Simonis, 2012], the user provides an initial set of positive and (possibly) negative configurations from which constraints are mined. *Active* constraint acquisition methods, on the other hand, iteratively query the user by asking to classify full [Bessiere *et al.*, 2007] or partial [Bessiere *et al.*, 2013] assignments as positive or negative examples, or to provide constraints violated by a negative example [Freuder and Wallace, 1998].

Acknowledgments ST was supported by the Caritro Foundation through project E62I15000530007.

References

[Beldiceanu and Simonis, 2012] N. Beldiceanu and H. Simonis. A model seeker: Extracting global constraint models from positive examples. In *CP'12*, pages 141–157, 2012.

[Bessiere *et al.*, 2005] C. Bessiere, R. Coletta, F. Koriche, and B. O'Sullivan. A sat-based version space algorithm for acquiring constraint satisfaction problems. In *Proceedings of ECML*, pages 23–34, 2005.

[Bessiere *et al.*, 2007] C. Bessiere, R. Coletta, B. O'Sullivan, and M. Paulin. Query-driven constraint acquisition. In *IJCAI'07*, pages 50–55, 2007.

[Bessiere *et al.*, 2013] C. Bessiere, R. Coletta, E. Hebrard, G. Katsirelos, N. Lazaar, N. Narodytska, C-G. Quimper, and T. Walsh. Constraint acquisition via partial queries. In *IJCAI'13*, pages 475–481, 2013.

[Boutilier *et al.*, 2004] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *JAIR*, 21:135–191, 2004.

[Chajewska *et al.*, 2000] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *AAAI/IAAI*, pages 363–369, 2000.

[Chen and Pu, 2012] L. Chen and P. Pu. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1-2):125–150, 2012.

[Freuder and Wallace, 1998] E. C. Freuder and R. J. Wallace. Suggestion strategies for constraint-based matchmaker agents. In *CP'98*, pages 192–204, 1998.

[Goetschalckx *et al.*, 2015] R. Goetschalckx, A. Fern, and P. Tadepalli. Multitask coactive learning. In *IJCAI'14*, pages 3518–3524, 2015.

[Goldsmith and Junker, 2009] J. Goldsmith and U. Junker. Preference handling for artificial intelligence. *AI Magazine*, 29(4):9, 2009.

[Guerin *et al.*, 2013] J. T. Guerin, T. E. Allen, and J. Goldsmith. Learning cp-net preferences online from user queries. In *Algorithmic Decision Theory*, pages 208–220. 2013.

[Koriche and Zanuttini, 2010] F. Koriche and B. Zanuttini. Learning conditional preference networks. *AI*, 174(11):685–703, 2010.

[Lallouet *et al.*, 2010] A. Lallouet, M. Lopez, L. Martin, and C. Vrain. On Learning Constraint Problems. In *ICTAI'10*, pages 45–52, October 2010.

[Raman *et al.*, 2013] K. Raman, T. Joachims, P. Shivaswamy, and T. Schnabel. Stable coactive learning via perturbation. In *ICML'13*, pages 837–845, 2013.

[Sebastiani and Tomasi, 2015] R. Sebastiani and S. Tomasi. Optimization modulo theories with linear rational costs. *ACM TOCL*, 16(2):12, 2015.

[Shivaswamy and Joachims, 2015] P. Shivaswamy and T. Joachims. Coactive learning. *JAIR*, 53(1):1–40, 2015.

[Teso *et al.*, 2016] S. Teso, A. Passerini, and P. Viappiani. Constructive preference elicitation by setwise max-margin learning. In *IJCAI'16*, 2016. (To appear).

[Viappiani *et al.*, 2006] P. Viappiani, B. Faltings, and P. Pu. Preference-based search using example-critiquing with suggestions. *JAIR*, pages 465–503, 2006.

[Wolsey and Nemhauser, 2014] L. Wolsey and George L. Nemhauser. *Integer and combinatorial optimization*. 2014.